Realistic 3D simulation of shapes and shadows for image processing

Jean-Philippe Thirion

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 1344

Programme 6
Robotique, Image et Vision

REALISTIC 3D SIMULATION OF SHAPES AND SHADOWS FOR IMAGE PROCESSING

Jean-Philippe THIRION

Novembre 1990

RR-1344

Realistic 3D Simulation of Shapes and Shadows for Image Processing

Simulation réaliste des formes et des ombres pour le traitement d'images

Programme 4: Robotique, Image et Vision

Jean-Philippe THIRION [1]

INRIA, Domaine de Voluceau, Rocquencourt
B.P. 105, 78153 Le Chesnay Cedex
FRANCE
Email: thirion@bora.inria.fr
Tel: (33 1) 39 63 52 79
Fax: (33 1) 39 63 53 30

November 19,1990

[1] is now a research member of the Epidaure project at INRIA Rocquencourt, France, but parts of this work have been performed when the author worked at the LIENS , Paris, France.

Abstract

This paper illustrates the cooperation between Image Processing and Computer Graphics. We present a new method to compute realistic 3D images of buildings or complex objects from a set of real pictures and from the 3D model of a real scene. We also show how to remove real shadows from those pictures and how to simulate new lightnings. Our system allows the generation of synthetic pictures, with a total control over the position of the camera, over the features of the optical system, and over the solar lightning. We propose several methods to avoid most of the artefact which would be produced by a straitforward application of our approach. At last, we propose a general scheme to use these pictures in order to design new optical systems and to test Image Processing algorithms, long before the building of the first physical prototype.

Résumé

Cet article décrit la complémentarité de l'analyse et de la synthèse d'images. Nous présentons ici un nouveau moyen de synthétiser des images réalistes de bâtiments ou d'objets complexes à l'aide d'un jeu de photographies réelles et d'un modèle en trois dimensions du site concerné. Nous montrons également comment retirer les ombres des images réelles, et comment en rajouter de nouvelles. Notre système permet la production d'images synthétiques avec un controle total de la position de la caméra, des caractéristiques du système optique, et de l'éclairage solaire. L'utilisation directe de notre algorithme produit un certain nombre d'artefacts, et nous proposons plusieurs méthodes afin déviter la plupart de ceux-ci. Enfin, nous décrivons un schéma général d'utilisation de ces images pour la conception de systèmes optiques et la mise au point des algorithmes de traitement d'images, bien avant la construction du premier prototype physique.

1 Introduction

The scientific approach has been centered for a long time upon a loop between theory and experiments. First, the researcher look at the real world, and settle a theoretical model that allows him to make some predictions. He performs experiments and compares the results with the predictions. Depending on the difference between the theory and the reality, the researcher changes the theoretical model or improves it, and goes back to the first step. But real experiments are often very expensive to perform and many parameters are not well mastered. This is why simulation has been raised to a completely new field in science during those last years. In fact, simulation have always been part of the scientific approach. Making predictions from a theoretical model is some kind of simulation, whether those predictions are hand-made or computed. Because computers become more and more powerful each days, the theory-experiment loop turns into a theory-simulation loop, with sometimes registration to the reality by the way of real experiments.

Simulations have been used by Image Processing researchers for a long time, but simulated images have mainly been produced with two dimensional techniques, in order to test two dimensional algorithms. The third dimension is much more expensive, and only recent progresses in Computer Graphics allow those kind of simulations.

This is why a cooperation between Computer Graphics and Image Processing is so fruitful nowadays. Several recent works emphasize this cooperation between analysis and synthesis for solving 3D problems, such as in [7] for the analysis of shape from motion, or as in [4] for robotic applications.

We present in section 2. a way to use Image Synthesis techniques in order to simulate realistic pictures. One original point is that our method deals with the true 3D models of buildings, or of complex objects and not only with terrain modeling ($z = f(x, y)$). This approach is close to [3], but allows to modelize any kind of camera. In short, we start from a 3D model of a real scene, and a set of real pictures and we solve the hidden surface problems in order to find the real texture of the objects and to map this texture onto the 3D model. This allows us to compute any view of the scene, with a complete hold over the geometrical parameters, as for example the camera position, or the shape of the objects. We can even simulate a complete movie with a total control over the trajectory of the camera. We use ray tracing to compute the synthetic images, so we are able to modelize complex cameras, or any failure of the optical system. The method we present in this paper is very useful to test the image processing algorithms which are used, as for example, to correct optical failures, or to settle the shape of objects (see [5], [8]). Comparisons with the initial shapes will prove how effective the image

processing algorithms are.

But sometimes, Image Processing needs more. We present in section 3. a way to remove the real shadows of the real pictures, and how to compute synthetic shadows for those pictures. The two techniques can be used at the same time to produce images with a complete hold on both geometric and lightning features. Those images will be very useful to test algorithms dealing with shadows, for example in order to extract the shapes of the objects from their shadows [8].

In section 4. we propose a general scheme to use those synthetic images in order to improve both the synthesis and analysis processes.

2 Geometric simulation

Let us focus on the geometric simulation of the scene. We suppose that we have a three dimensional model of real buildings, and a set of real pictures. How to get the 3D model is not in the scope of this paper. One can think of making direct measurement of the buildings or of the objects, or using the drafts of construction. In our case, 3D models have been built from stereoscopic views of real scenes (with the TRAPU acquisition system of the IGN). This is a particularly interesting method for us because those real pictures are used both for extracting the 3D model and for synthetizing new images. We possess also the position and the characteristics of the camera for each pictures. For rendering purpose, the scene was decomposed into elementary triangles.

2.1 Hidden surfaces for the real images

On a given picture, there are many parts of the scene which remain hidden. Image Synthesis researchers have classified several types of hidden surface problems and brought solutions to them (see [10]):

- **Clipping:** The corresponding technique settles which parts of the objects in a scene are in the cone of vision, starting from the camera. The objects outside this cone cannot be seen on the picture (if we suppose there is no reflections).

- **Back-Face Culling:** For closed objects, a single test with the normal orientation, according to the position of the object and the position of the viewer allows to discard approximately a half of the surface elements. Only the front facing elements are potentially visible on the image.

- **Direct Hiding:** For a given point of view, a surface may geometrically hide another surface. Many solutions have been provided, among them, the **z-buffer** algorithm and the **ray tracing** algorithm.

Because our cameras are well approximated with a conic perspective, we choose to use a z-buffer algorithm to solve hidden surfaces problems in the real image space: the front facing elements of the scene can be projected onto the image plane, where we perform a 2D clipping and solve the hidden surfaces problems for each pixel, according to the distance between the camera and the objects. We build then an **Item Buffer** that contains, for each pixel of the picture, a reference to the element visible through this pixel, or 0 if no elements are visible (introduced in [12]). This item buffer can be used to compute a synthetic image and this allows us to check the correctness of our 3D model and camera position. The synthetized image and the real picture must fit exactly. In fact, there is always a slight difference, about the size of a pixel, which is due to the discrete sampling of the scene. Those aliasing problems provide artefacts, as we will see later. An important point is that the computation of the item buffers for the real images can be performed once for all, as a preprocessing for any further simulations.

2.2 Hidden surfaces for the synthetic images

We want to synthetize any possible view of the real scene, therefore we are once again obliged to solve the hidden surfaces problems for new points of views. We can use the same technic as before if we want to modelize a virtual camera with a conic perspective. But we can also discretize the behavior of the optical system by the way of **ray casting** ([11]). For each pixel of the synthetic image, we cast a ray through that pixel and toward the scene, and compute the first intersected object. This allows us to deal with any type of camera, like for example complex CCD cameras. We are also able to modelize the failures of the optical system when disturbing the rays.

2.3 Mapping of the real texture

We are now able to map the texture of the real picture onto the synthetic image. For each pixel of the synthetic image, ray tracing settles the first intersected object and the point of intersection. This point is projected back onto the real image, and this projection corresponds to non-integer coordinates into the real picture (see figure 1). The item buffer associated with the real picture settles if the object, visible at this point, corresponds to the same object as the one intersected by the ray, in the synthetic image. If it

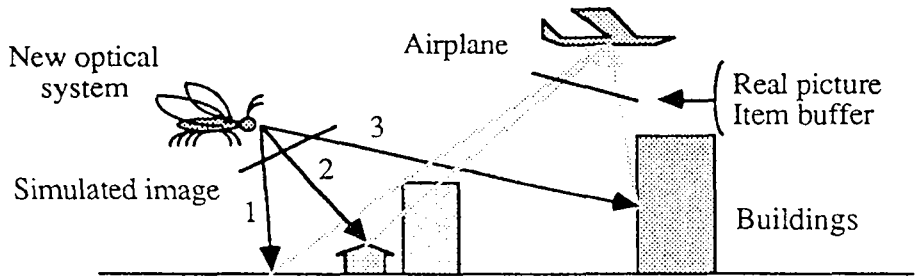

Figure 1: Finding the texture in the real pictures for a given point

is the same object, then the texture in the real image is linearly interpolated and given to the synthetized image.

For example, ray 2 in figure 1 corresponds to a point of the item buffer, but the element number of the intersected surface and of the item buffer does not match, therefore this picture is not used.

The ray tracing process is sometimes very slow, but, as we can see, the mapping of the real texture is very simple. If we use z-buffer instead of ray tracing, a real time synthetizer of realistic images can be foreseen. Only the large amount of memory required to maintain the real pictures and the item buffers prevent this technic from being directly used for air flight simulation. To illustrate the method, picture 9 is one of the two original pictures (the closest to the synthetic view) used to synthetize the new image of figure 10.

2.4 Artefacts produced with this method

Several types of artefacts may appear during this process, we have classified those artefacts and tried to solve most of them. These are:

- **No visible surfaces :** Because we simulate any view of the real scene, there are points were no information is available on any of the real pictures.

- **Tangential viewing :** A surface might be front viewed on the synthetic image and might be seen tangentially on the real pictures. Therefore, we have some information about the texture at this point, but this information is very 'diluted'. On the synthetic image, the real texture is 'stretched' (ray 3 in figure 1). This can be seen on the sides of the building of Jussieu, in image 10.

- **Aliasing problems :** We synthetize a discrete image, so aliasing errors appear, producing artefacts as "stairs" along the edges of the buildings.

Adaptativ supersampling or distributed ray tracing ([2]) reduce those artefacts (but are not used in this paper). Unfortunately, aliasing appears also on the item buffers associated to the real images. For a frontal surface in the synthetized image, and a tangential view in the real image, these artefacts will be emphasized.

- **Errors of the 3D model :** Any slight error of the 3D model leads to a bad association of the surfaces with the real picture. Some pixels are wrongly associated to a surface. For example, suppose that we have to modelize a scene composed with a floor and a building. When simulating the image, if a ray intersects the floor at a point corresponding to the limit between the floor and the top of a building on the real picture, then a slight difference between the item buffer and the real picture, due to a modeling error or even to an aliasing error, will associate the color of the roof to this pixel instead of the color of the floor (see ray 1 in figure 1). This will produce a 'ghost' edge of the roof on the floor of the synthetic image.

- **Lightning changes :** Lightning may change from a real picture to the other. If we choose to map the texture coming from a real picture to a given pixel, and the texture coming from another picture to the next, the difference may appear in the synthetic image.

2.5 Some solutions to those problems

First of all, we equalize the intensity of all the images we use, in order to lessen the lightning changes between pictures. Then, for each pixel of the synthetic image, we choose between all the real pictures the most suitable one. We present here some of the criteria that we use to select the best picture :

- **Priority :** We set a priority among the real pictures depending on the viewing angle. The image with the closest direction of viewing to the synthetized image will have the highest priority. This will produce less changes from pixel to pixel, because we use almost always the same picture, this also lessen the lightning changes. It is a preprocessing for the whole simulation.

- **Orientation :** We use the orientation of the surface, in order to avoid tangential viewing, and select the most frontal view for this surface.

- **Neighbourhood :** We study the neighbours of the selected pixel in the real picture. If they all correspond to the same surface as the selected

pixel one, then there is less chance to perform a false association of the texture, due to aliasing or geometrical errors.

Those criteria must be weighted depending on the kind of simulation. They gave us satisfying results, but other criteria may be foreseen. There are also some cases where the information is lacking. We compute an average intensity for all the pictures and provide it when no other information is available. We can also easely compute, with the real pictures and the item buffers, an average intensity for surface elements, when parts of those elements are visible, and affect this intensity to hidden parts of those surfaces.

In general, with several views of the scene (a top view and four side views), and with supersampling for both the item buffers and the simulated image, the resulting picture is very satisfying. Unfortunately we had only vertical views to perform our simulations, and we used the spread angle of the camera for side views of the buildings. We used also almost no supersampling for the synthetic images (our synthetic images have about the same resolution as the original pictures). But nevertheless the result is not too bad (see images 9 and 10).

3 Shadow simulations

We will see in this section how to remove shadows from the real pictures and how to add new ones. Dealing with this problem may seem very strange, but many informations about the shape of the objects can be extracted from their shadows. Taking the same picture of a real scene with exactly the same geometrical characteristics but with different lightnings may be very difficult and expensive (think of aerial views for example). Our system allow the test of many algorithms dealing with shadows, with the possibility to define any type of solar lightnings, including boreal lightning, and with a total control on the geometrical characteristics.

3.1 Lightning model

The image synthesis field has proposed a number of increasingly accurate models for illumination. Figure 2 describes the Phong lightning model which takes both diffuse and specular aspects into account (see [9]). A more complete equation has been proposed by Kajiya, in [6]. In the present paper, we used only a direct lightning model. We do not modelize the diffusion of the light from surfaces to surfaces (figure 3), neither the specular reflection of light.

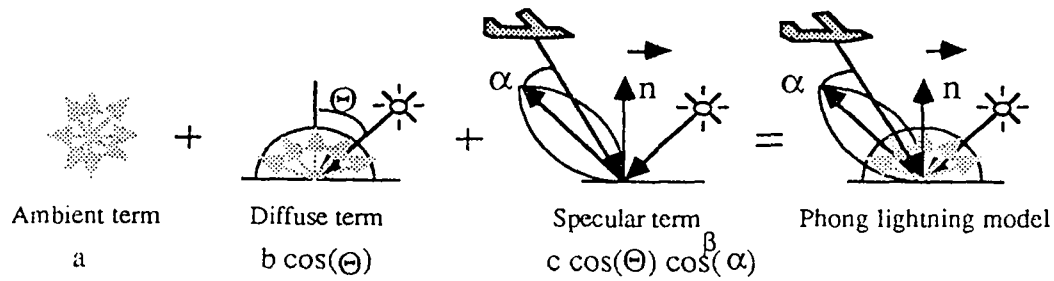

Figure 2: The Phong lightning model

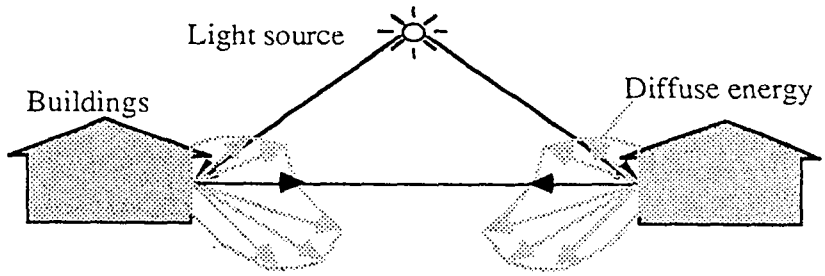

Figure 3: Radiosity: Diffusion of light from surfaces to surfaces

The use of a radiosity method (see [1]) may lead to more precise results. We use a much more simple model defined by equation 1 :

$$Intensity = [a + b \cdot \delta \cdot \cos(\Theta)] \cdot Albedo \tag{1}$$

with :

Intensity the isotropic re-emited intensity at a point.

Albedo the albedo of the surface at this point.

a the diffuse energy at this point.

b the intensity of the solar lightning.

δ is 0 or 1 depending on the visibility from the light source.

Θ the angle between the incident light and the surface normal

We make the crude assumption that the diffuse compound a is constant for the whole scene. We also suppose that we have no information about the acquisition, development and digitalization processing of the real picture. Our goal is not to build a complex lightning model and to extract the physical intensity and albedo of the surfaces, but to solve the geometrical problems which appear when simulating lightning for images, and to keep on using real textures.

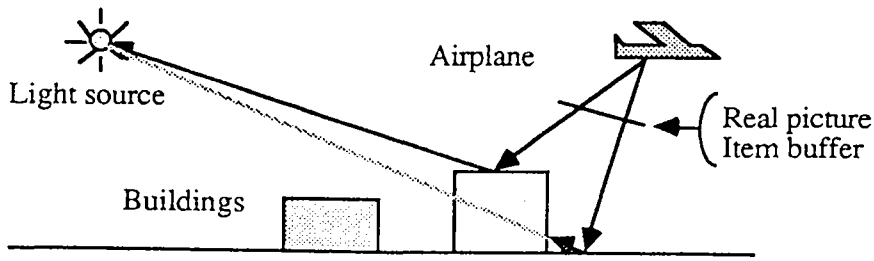

Figure 4: Building the shadow buffer

3.2 Finding shadows

As for the visibility, there are several kinds of shadows. First, the back facing surfaces, with respect to the position of the light source, are in their own shadow, and are easy to settle. There are also cast shadows, from object to object. These shadows are settled with ray tracing : for each point corresponding to a pixel of the real picture, we cast a ray toward the light source. If this ray is intercepted, then the point is in the shade. We build an image, that we call the **shading buffer**, with '0' when the point is entirely in the shadow, with '1' if there is no object of the model for this point, and with the cosine between the incident light and the surface normal ($\delta \cdot \cos(\Theta)$) anywhere else (see figure 4). If we do not know at what time the real photographs were taken, the position of the sun can be directly measured onto the pictures, with the corner of buildings and the corresponding shadows. This is possible because we have a precise 3D model of the scene. Picture 12 is a real picture, the image 13 corresponds to the shading buffer of this picture.

3.3 Removing shadows from real pictures

What we are interested in is to find the albedo of the surface. Because we know nothing about the development process of the real picture, we can't find the physical albedo of those surfaces. Furthermore, we have not the physical intensity at each pixel, but a grey level I that comes through the acquisition, development and digitalization processing. Therefore, we look for a pseudo-albedo A and make the assumption that equation 1 is satisfied. If we invert this formula : $A = I/(a + b \cdot \delta \cdot \cos(\Theta))$. At the end of the process, we want to have a digital image, that we call the **pseudo-albedo image**, with approximately the same range of values than the original picture. We found the following criterion a satisfying normalization condition.

$A = I$ for all the sunlit horizontal surfaces of the scene

This formula as the advantage to let most of the surfaces of the real pictures unchanged. If $\Theta 1$ is the angle between the incident light and the horizontal plane, we have the relation 2 between a and b.

$$a + b \cdot \cos(\Theta 1) = 1 \tag{2}$$

We need a second relation in order to settle coefficients a and b. We get this relation with a measurement on the real picture. Suppose that we have an horizontal surface with a constant pseudo-albedo A1. Assume that there is a sunny and a shady part for this surface, with respectively the intensity I_{sun} and I_{shadow}. We have then :

$$I_{sun} = (a + b \cdot \cos(\Theta 1)) \cdot A1 \tag{3}$$

$$I_{shadow} = a \cdot A1 \tag{4}$$

Therefore:

$$a = \frac{I_{shadow}}{I_{sun}} \tag{5}$$

$$b = \frac{1 - (I_{shadow}/I_{sun})}{\cos(\Theta 1)} \tag{6}$$

This model works for many parts of the scene, but unfortunately, there are parts where the fraction I_{shadow}/I_{sun} is far from being constant. This is due to many different things. First, the acquisition, development and digitalization processes are logarithmical rather than linear. Second, there are radiosity effects that locally disturb the value of the coheficient a, and that we do not modelize. We experimentally settled that those disturbances appears more often in the shady parts of the scene. We set a different formula for those surfaces. We will suppose that for the shadow points, the intensity depends on the albedo according to an independent formula. High albedo surfaces generaly enhance the local ambient light, and conversely, low albedo reduce the ambient light (but the reality is much more complicated, because of geometric factors). We choose to take the linear formula 7 to modelize this phenomenon, with the same normalization as for the horizontal surfaces of the scene, ($A = I_{sun}$):

$$I_{shadow} = u + v \cdot A \tag{7}$$

To summarize the process, we measure two couples of intensity for two sunny and shady horizontal surfaces, $(I_{shadow1}, I_{sun1})$ and $(I_{shadow2}, I_{sun2})$ corresponding to respectively a high intensity and a low intensity, in order to set the coefficients a and b of equations 5 and 6, and coefficients u and v of

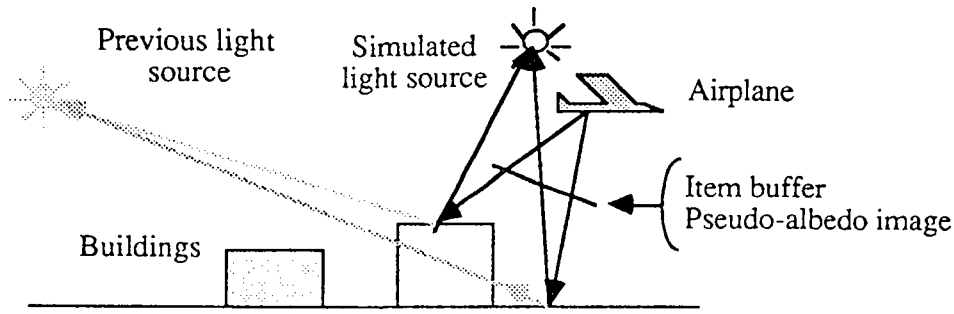

Figure 5: Simulate new shadows on a real image

equations 8 and 9. Image 14 is the pseudo-albedo image of picture 12.

$$v = \frac{(I_{shadow1}/I_{sun1}) - (I_{shadow2}/I_{sun2})}{I_{shadow1} - I_{sun2}} \tag{8}$$

$$u = \frac{I_{shadow1}}{I_{sun1} - I_{shadow1} \cdot v} \tag{9}$$

3.4 Adding new shadows on real pictures

The formula 1 and 7 are used to preprocess the real pictures and to transform them into pseudo-albedo images, respectively for sunny and shady surfaces. Because we do not simulate radiosity phenomena, only formula 1 will be used to simulate new lightnings on both sunny and shady surfaces. Dealing with coefficient a and coheficient b will allow us to change the lightning conditions of the scene during the simulating process. The geometric position of new shadows is settled with the same method as for removing shadows. We build a shadow buffer with ray tracing, with $\delta \cdot \cos(\Theta)$ for each pixels, and we use formula 1 to compute the simulated image (see figure 5).

In the images that we present in this paper, we use a low a coheficient, in order to emphasize the new shadows. But we can change those coefficients in order to produce less sunny days, ranging from the completly diffuse lightning of the pseudo-albedo image 14, to the sunny evening of the simulated image 15. A second point is that the computation of a synthetic image with the same coefficients a and b, and with the same position of the sun leads to almost the same image as the real picture. The slight difference is due to radiosity effects. This is used as a test for our method.

3.5 Artefacts for shadow simulations

There are several causes for the artefacts in the simulated images:

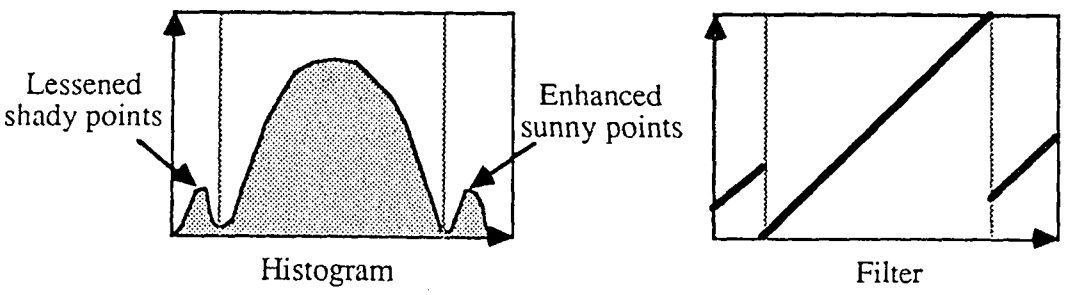

Figure 6: Filtering the out of range points

- **Aliasing errors :** We sample the shadow informations with rays, thus there are traditional aliasing errors. Stairs will appear on the edges of the simulated shadows of the buildings. Once again, adaptativ supersampling will help to reduce those artefacts.

- **Modeling errors :** During the preprocessing of the image, the slightest error on the model, or even aliasing errors, will produce differences between the real picture and the shadow buffer. Therefore, some shady points are settled to be sunny, and their low intensity is reduced, and conversely, some sunny points are settled to be shady, and their high intensity is enhanced. This effect can be seen on the image 14. There is an overhang of the roof of buildings that have not being modelized. Thus, the shadows of the edges of those buildings are not correctly computed, and sunny points are enhanced, and can be seen as little white vertical segments starting from the roof. We use a very simple algorithm to reduce those effects. Because the modeling errors make the algorithm performing exactly the opposite of what it is supposed to do, the albedo of those points goes out of the range of the normal spectrum of the histogram. Thus it is easy to find these points and bring some correction, with a filter (see figure 6).

- **Noise, radiosity, ...** Once again, we do not modelize radiosity phenomena. This leads to color differences, even on surfaces with a constant albedo. A close look to the simulated images may reveal the old positions of the shadows. Differences may also appears for particularly dark places, that are enlightened on the simulated images. As the intensity of the image is enhanced for those points, the noise of the image is enhanced too, and this difference of noise may also reveal the previous position of shadows.

Despite of all these artefacts, the synthetized images remain a good first order approximation of the new lightning of the scene. Supersampling, and

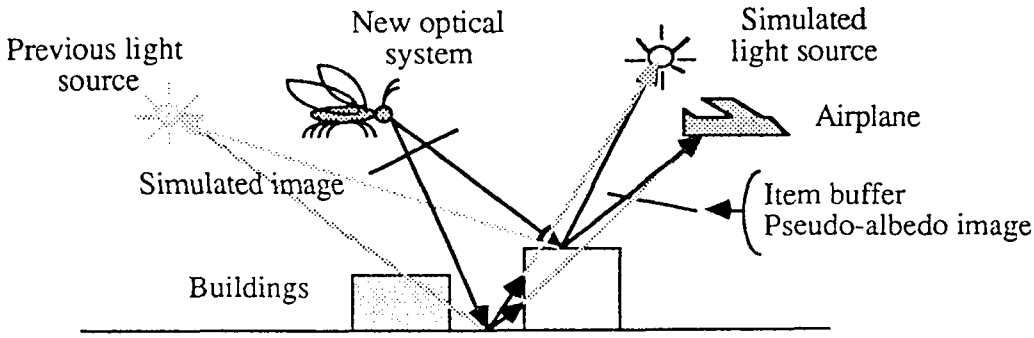

Figure 7: Simulate a new view with a new lightning

the use of radiosity technics, may help to simulate more precise picture (we do not use supersampling in this paper in order to show artefacts, and radiosity leads to much more complicated implementations ...).

3.6 Merging geometric and shadow simulations

In order to simulate any view with any lightning of the scene, we do the following :

- **Preprocessing :** for all the real pictures.
  1. Compute the item buffer associated to the real picture.
  2. Measure the position of the sun and several couples of intensity
  3. Compute the shadow buffer associated to the picture.
  4. Compute the pseudo-albedo image, and filter this image in order to remove modeling and aliasing errors.
- **Processing** for all the simulated images, and for all the pixels of those images :
  1. Compute the ray, and the first intersection between the ray and the 3D model. Send a ray from this point toward the sun.
  2. Interpolate the color of the point into one of the pseudo-albedo image (whereas we used the real pictures for geometrical simulations in the first section of this paper).
  3. Apply function 1 to this point.

Figure 7 summarize this process. Image 11 is a simulated view of a building, with a simulated lightning.

4 The use of realistic simulations

There are many possible uses for those simulated pictures. We present here a general scheme to design a new optical system, to test it, and to improve its performances. The previous sections may be considered as an application of the scheme that we describe now. Other types of modeling and lightning systems may be foreseen, for other purposes than airplane applications.

4.1 Design of the optical system

When an optical system is designed, many questions are raised, such as :

- What will be the performances of this optical system ?
- How can I improve those performances ?
- Will the customers of this optical system be satisfied with the quality of the images ?
- Will my image processing algorithms work well with the images that the optical system will produce ?
- ...

And so on ... There is a need to exactly simulate the images that the optical system will take, before starting to build the first prototype. Depending on the optical system, it is sometimes possible to simulate those images with 2D technics. But in many cases the artefacts are related to three dimensional geometrical problems. For example let us consider a pushbroom airplane acquisition system. In this system, a line of CCD detectors 'sweeps' a given field along the time. Artefacts appears because of failures of the optical system (wrong position of CCD detectors in the focal plane), and also because of the trajectory errors of the air plane. The artefacts depend on the nature of the site, and will be different for plane fields, mountains, or city buildings. Will a shape extracting algorithm work on such an image ?

We start from a set of real images, extract the 3D models of those scenes. We can then simulate images according to our knowledge of the future optical system, and use image processing algorithm, for example to extract 3D models .... If we compare the 3D model coming from real pictures to the 3D model used to simulate pictures, we are able to settle the performances of our image processing algorithm, a long time before the first real prototype is built. Such a loop between the simulating system and the analyse algorithm is shown in figure 8. We may compare the simulated image to the real images

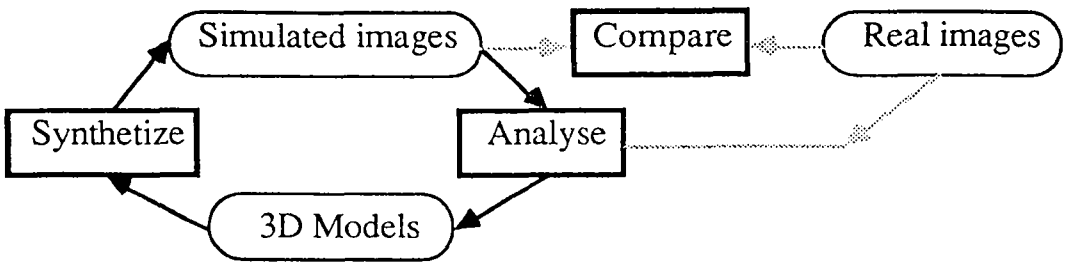

Figure 8: Loop between Analysis and Synthesis with registration to reality

and settle the performances of the optical system. We are also able to show the result to the users and improve the design of the optical system.

4.2 Improving the optical system

The simulating system may be used even when the first real prototype is built. Sometimes, some parameters have to be settled only when the optical system is working, (focusing, think about the Hubble telescope for example). Because it provides a direct access to many geometrical parameters, the simulating system may help to understand what is happening, and how to set the parameters, in order to produce best pictures.

4.3 Improving the simulating process

With this first prototype, we get real images that may be compared with simulated images. The differences will come from modeling errors, or unforeseen events, and determining where errors occur will be very fruitful : a phenomenon may have been underestimated or ignored when designing the optical system, and this information will be very important both for the next generation of the optical system, and for improving the simulating process.

5 Conclusion

We have presented in this paper a way to simulate realistic images, with realistic shadows, in order to produce images and to test image processing algorithms a long time before the first prototype of a new optical system is built. The lightning model that we use is very basic, but leads to good results, further work might be to take radiosity into account. Adaptativ supersampling may also be used at several level in our simulating system in order to reduce aliasing artefacts. We hope that computer graphic technics

will help to design more complex optical systems, and image processing researchers to build more powerful algorithms, dealing with 3D geometry.

Acknowledgment

Several French companies and state organisms have directly or indirectly participated to this work. I wish particularly to thank Mr. Eric Savaria and Mr. Roberto Aloisi from the French Aérospatiale, Mr. Jean-Paul Arif from Matra MS2I who provided the 3D model of Jussieu, Mr Philippe Campagne from the Institut Géographique National (IGN France), who provided the pictures and some the 3D models, and Mr Jean-Michel Lachiver from the Centre National d'Etudes Spatiales (CNES) .

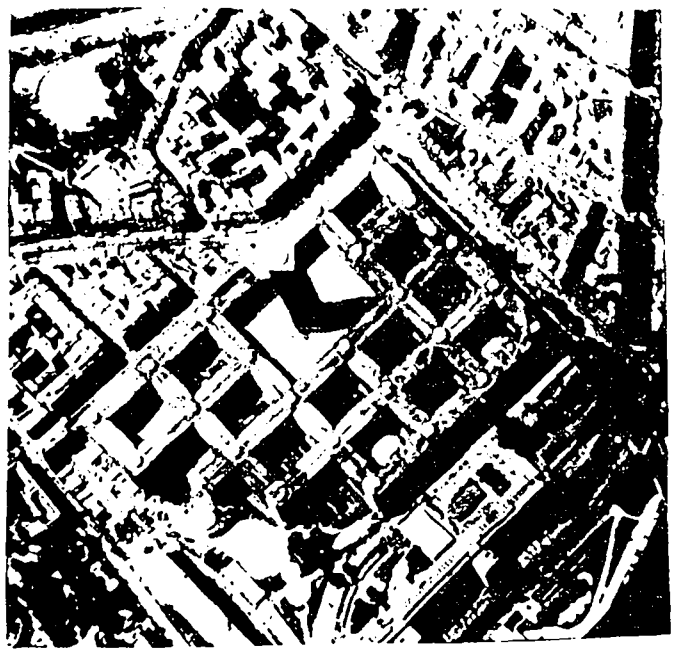

Figure 9: One of the two original real pictures of the University of Jussieu (the closest to the next simulated image).

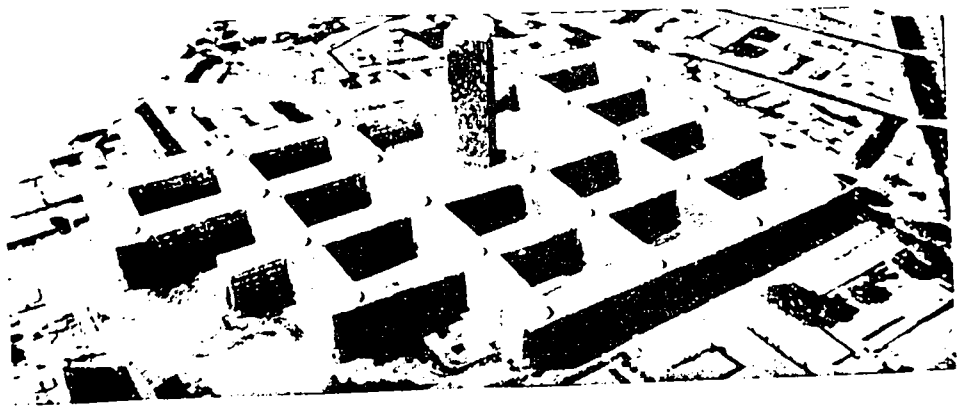

Figure 10: The simulated image of Jussieu. Note that the other buildings, that have not been modelized, appear 'flat'.

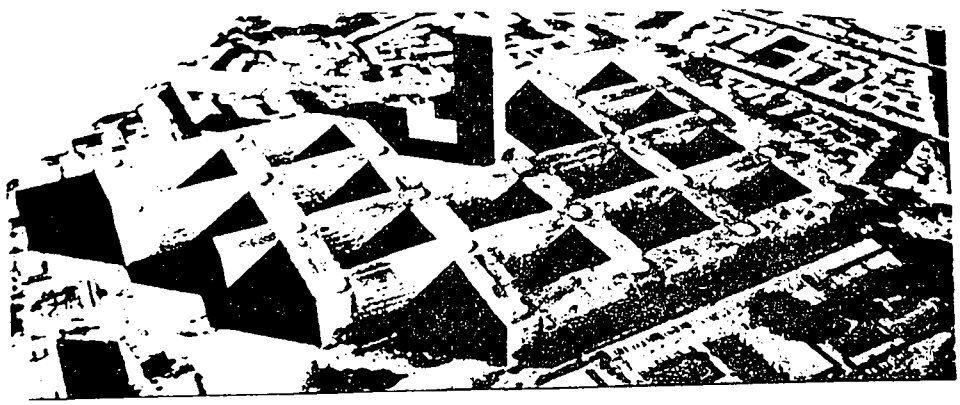

Figure 11: The simulated image of Jussieu with a synthetic lightning.

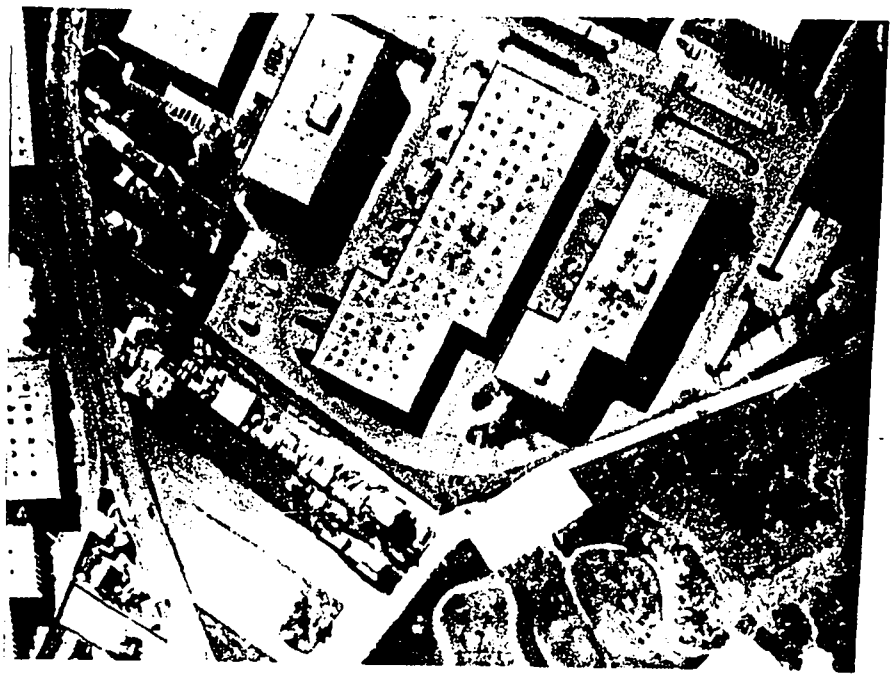

Figure 12: A real picture of Rungis.

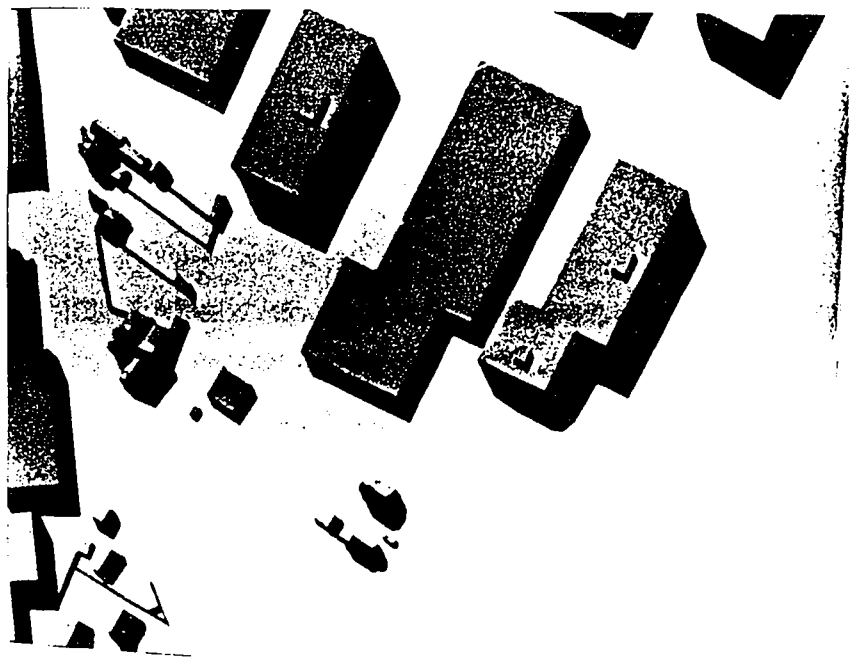

Figure 13: The shadow buffer associated to the picture of Rungis.

Figure 14: The pseudo-albedo image associated to the picture of Rungis.

Figure 15: The synthetic view of Rungis with a new lightning.

References

[1] Michael F. Cohen and Donald P. Greenberg. The hemi-cube : A radiosity solution for complex environments. *Computer Graphics*, 19(3):31–40, July 1985. Proceedings of the ACM SIGGRAPH.

[2] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *Computer Graphics*, 18(3):137–145, July 1984.

[3] Robert N. Devich and Frederick M. Weinhaus. Image perspective transformations. *SPIE vol.238 Image Processing for Missile Guidance*; pages 322–332, 1980.

[4] A. Gagalowicz. Collaboration between computer graphics and computer vision. *Scientific Visualization and Graphics Simulation*, pages 233–248, 1990.

[5] A. Huertas and R. Nevatia. Detecting buildings in aerial images. *Comp. Vision, Graphics and Image Process.*, pages 131–152, 1988.

[6] James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, August 1986. Proceedings of the ACM SIGGRAPH.

[7] Reinhard Koch. Automatic modelling of natural scenes for generating synthetic movies. *Proceedings of EUROGRAPHICS'90*, pages 215–224, September 1990.

[8] Yuh-Tay Liow and Theo Pavlidis. Use of shadows for extracting buildings in aerial images. *Comp. Vision, Graphics and Image Process.*, pages 242–277, 1990.

[9] B. Phong. Illumination for computer genereted pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[10] I.E. Sutherland, R.F. Sproull, and A. Schumacker. A characterization of ten hidden-surface algorithms. *Computing Surveys*, 6:1–25, 1974.

[11] Whitted T. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.

[12] Hank Weghorst, Gary Hooper, and Donald P. Greenberg. Improved computational methods for ray tracing. *ACM Transactions on Graphics*, 3(1):52–69, January 1984.

Imprimé en France
par
. l'Institut National de Recherche en Informatique et en Automatique

ISSN 0249-6399

ISSN 0249-6399